



# FACULTAD DE INGENIERIA

Universidad de Buenos Aires

## CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

**Realidad aumentada para la  
optimización de procesos industriales**

**Autor:**  
**Iván Szkrabko**

Director:  
Mg. Ing. Leandro Lanzieri (FIUBA)

Jurados:  
Esp. Ing. Daniel Marquez (FIUBA)  
Mg. Ing. Roberto Compañy (FIUBA)  
Mg. Ing. Matías Álvarez (FIUBA)

*Este trabajo fue realizado en la Ciudad Autónoma de Buenos Aires,  
entre Junio de 2020 y Junio de 2021.*



## *Resumen*

En este trabajo se presenta el desarrollo de una aplicación de realidad aumentada para uso industrial. El trabajo surge como una prueba de concepto para la empresa ABB. La aplicación se desarrolló con el fin de mejorar el desempeño de los operadores en la ejecución de procedimientos industriales, típicos en plantas de lubricantes o farmacéuticas, donde la producción tiene que cumplir con un estricto control de calidad.

Para la ejecución del trabajo se utilizaron conceptos de programación orientada a objetos, testing y programación multihilos. También se aplicaron conocimientos referentes al desarrollo de endpoints para la API y gestión de bases de datos. Por otro lado, gracias a la gestión de proyectos, se logró una buena comunicación con los interesados y llevar el control a lo largo de todo el desarrollo.



## *Agradecimientos*

Agradezco a mi Director por su apoyo a la investigación y el desarrollo, y a mis Jurados por la evaluación de la presente memoria.



# Índice general

<b>Resumen</b>	<b>I</b>
<b>1. Introducción general</b>	<b>1</b>
1.1. Realidad aumentada . . . . .	1
1.2. Aplicaciones de realidad aumentada en la industria . . . . .	2
<i>Smart retrofitting in the context of industry 4.0</i> . . . . .	2
<i>Production workplace enhanced by mixed reality</i> . . . . .	2
<i>Mixed reality technology for onsite construction assembly</i> . . . . .	2
1.3. Procedimientos <i>batch</i> . . . . .	2
1.4. DCS utilizado en el trabajo . . . . .	3
1.5. Motivación . . . . .	5
1.6. Alcance . . . . .	5
<b>2. Introducción específica</b>	<b>7</b>
2.1. Entorno de desarrollo . . . . .	7
2.1.1. Hololens2 . . . . .	7
2.1.2. MRTK . . . . .	8
2.1.3. OPC <i>framework</i> . . . . .	10
2.2. Requerimientos . . . . .	13
2.3. Planificación . . . . .	14
<b>3. Diseño e implementación</b>	<b>15</b>
3.1. Análisis del software . . . . .	15
3.1.1. Arquitectura del sistema . . . . .	15
3.1.2. Interfaz visual y lógica .NET . . . . .	16
3.1.3. API y base de datos . . . . .	18
3.1.4. Comunicación OPC . . . . .	19
3.1.5. Sistema de control . . . . .	20
<b>4. Ensayos y resultados</b>	<b>23</b>
4.1. Aplicación . . . . .	23
4.2. Evaluación de las interfaces . . . . .	27
4.3. Devoluciones del cliente . . . . .	28
<b>5. Conclusiones</b>	<b>29</b>
5.1. Resultados obtenidos . . . . .	29
5.2. Próximos pasos . . . . .	29
<b>Bibliografía</b>	<b>31</b>



# Índice de figuras

1.1. Clasificación de tecnologías <sup>1</sup> . . . . .	1
1.2. Proceso continuo (superior) vs proceso por lotes (inferior) . . . . .	3
1.3. Sistema de control distribuido 800xA <sup>2</sup> . . . . .	4
1.4. Controlador simulado <sup>3</sup> . . . . .	4
1.5. OPC server <sup>4</sup> . . . . .	5
2.1. Hololens 2 <sup>5</sup> . . . . .	7
2.2. Microespejos <sup>6</sup> . . . . .	8
2.3. Componentes del MRTK <i>framework</i> <sup>7</sup> . . . . .	9
2.4. Entorno de desarrollo <sup>8</sup> . . . . .	10
2.5. Arquitectura OPC <sup>9</sup> . . . . .	11
2.6. Diagrama en bloques <sup>10</sup> . . . . .	11
2.7. Objeto OPC <sup>11</sup> . . . . .	12
2.8. Diagrama de Gantt <sup>12</sup> . . . . .	14
3.1. Diagrama en bloques del sistema <sup>13</sup> . . . . .	15
3.2. Diagrama de flujo <sup>14</sup> . . . . .	16
3.3. Método de autenticación elegido <sup>15</sup> . . . . .	19
3.4. Bloque receta <sup>16</sup> . . . . .	20
3.5. Secuencia <sup>17</sup> . . . . .	21
4.1. Pantalla inicial <sup>18</sup> . . . . .	23
4.2. Código erróneo <sup>19</sup> . . . . .	24
4.3. Código correcto <sup>20</sup> . . . . .	24
4.4. Inicio de receta <sup>21</sup> . . . . .	24
4.5. Selección insumo <sup>22</sup> . . . . .	25
4.6. Insumo correcto <sup>23</sup> . . . . .	25
4.7. Producto elegido <sup>24</sup> . . . . .	26
4.8. Selección de producto <sup>25</sup> . . . . .	26
4.9. Fin de la aplicación <sup>26</sup> . . . . .	27



# Índice de tablas

2.1. Hardware Hololens 2 . . . . .	8
4.1. Tiempos de respuesta . . . . .	27



*Dedicado a mi familia*



# Capítulo 1

## Introducción general

En este capítulo se realiza una introducción a la realidad aumentada y los sistemas de control distribuidos. Además, se explica la motivación, alcance y objetivos del presente trabajo.

### 1.1. Realidad aumentada

Las tecnologías que permiten que un usuario visualice parte del mundo real a través de un dispositivo tecnológico se pueden clasificar en realidad virtual, realidad aumentada y realidad mixta. En la figura 1.1 se ilustran cada una de estas tecnologías.



FIGURA 1.1. Clasificación de tecnologías<sup>1</sup>.

La realidad virtual, consiste en un ambiente completamente digital. Se buscan aislar los sentidos del usuario del mundo real. De esta manera el usuario queda inmerso en otra realidad con la cual puede interactuar. Los dispositivos que permiten esta interacción son los cascos de realidad virtual y sus *joysticks*.

La realidad aumentada es una tecnología que permite conectar los dos mundos, el mundo físico y el mundo digital. Esto se logra superponiendo imágenes sobre la percepción que el usuario tiene del mundo real, mediante el uso de hologramas o incluso con teléfonos celulares. De esta forma en la pantalla se muestra no sólo el ambiente en el que está el usuario, sino que además se proyecta información adicional. Ejemplos de esta tecnología son los populares filtros de Instagram o el juego PokemonGo.

---

<sup>1</sup>Imagen tomada de <https://scand.com/company/blog/augmented-mixed-and-virtual-reality-for-business/>

Por último, la realidad mixta es una mejora de la realidad aumentada. Los elementos no sólo son adicionados sobre la percepción visual, sino que interactúan además con el espacio físico. Por ejemplo, si tuviéramos una pelota en una aplicación de realidad mixta, la misma podría rodar cuesta abajo de una escalera.

## 1.2. Aplicaciones de realidad aumentada en la industria

A continuación se presenta un listado de algunos trabajos contemporáneos relacionados con la realidad aumentada y su aplicación en la industria.

### *Smart retrofitting in the context of industry 4.0*

En este trabajo publicado en Elsevier [1], se utiliza el Hololens 2 para capacitar a los operadores luego de una mejora en una máquina dobladora de caños. El trabajo destaca que la herramienta permite a los operadores adaptarse rápidamente a la máquina, permitiéndoles trabajar de manera segura y guiada.

### *Production workplace enhanced by mixed reality*

En este paper presentado en la conferencia *Mensch und Computer* 2020 en Alemania [2], se muestra una aplicación de realidad aumentada para guiar al operador en el ensamblado de un scooter. El trabajo destaca el análisis de la relación costo-beneficio a la hora de armar procedimientos para los operadores. Plantean que deben encararse soluciones de realidad aumentada, donde el beneficio sea considerable, dado que estas aplicaciones requieren un elevado tiempo de desarrollo.

### *Mixed reality technology for onsite construction assembly*

En este paper presentado en la conferencia *Materials Science, Engineering and Chemistry* 2020 [3], se presenta una aplicación de realidad aumentada cuyo objetivo es comunicar los detalles de diseño para el montaje y la construcción. El trabajo destaca que el Hololens 2 es capaz de guiar la instalación de componentes eléctricos y de plomería, logrando un 9 % de mejora en la productividad. Menciona además, que las imágenes ocluidas en condiciones de luz solar, son el desafío a superar por el HoloLens 2 antes de una adopción generalizada.

## 1.3. Procedimientos *batch*

Un procedimiento *batch* o por lotes es aquel donde el producto sufre modificaciones consecutivas con el correr del tiempo[4]. El producto avanza por distintas etapas hasta estar terminado. En ese momento se evaca el producto de la instalación, se carga producto base y el ciclo comienza nuevamente. En la figura 1.2 se puede ver que en el proceso por lotes, sólo cuando se llega a  $t=4$  se obtiene el producto final deseado. Mientras que en el proceso continuo, el tanque contiene producto terminado desde el momento inicial  $t=0$ .

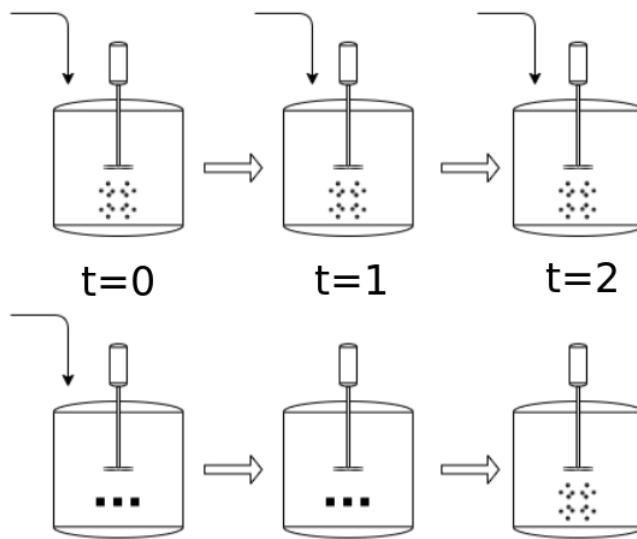


FIGURA 1.2. Proceso continuo (superior) vs proceso por lotes (inferior).

Algunas operaciones por lotes implican el uso de hojas de cálculo. Por ejemplo, un operador puede agregar una bolsa de químicos a un reactor y luego agregar un segundo químico. La razón entre el segundo y el primero debe ser exacta. Se necesita entonces una hoja de cálculo para determinar la cantidad requerida del segundo producto.

Los procedimientos operativos para los procesos por lotes son generalmente más complicados y difíciles de escribir que para las instalaciones de proceso continuo porque el tiempo es un factor. Además, muchas instalaciones están diseñadas para fabricar varios productos partiendo de las mismas máquinas en producción. Por lo tanto, se necesitan dos o más procedimientos para los mismos equipos de la planta, lo que da lugar a confusiones y malentendidos en los operadores.

## 1.4. DCS utilizado en el trabajo

Los sistemas de control distribuidos, también conocidos por sus siglas en inglés DCS [5], son sistemas compuestos por sensores, controladores y computadoras que se distribuyen por toda la planta. Cada uno de estos elementos tiene un propósito único, como la adquisición de datos, el control de procesos, el almacenamiento de datos o la visualización gráfica. Estos elementos individuales se comunican con un sistema centralizado a través de la red de área local de la planta, denominada red de control. En la figura 1.3, la arquitectura de red del sistema DCS de ABB [6], usado en este trabajo.

<sup>2</sup>Imagen tomada de <https://new.abb.com/control-systems/system-800xa/800xa-dcs/system/architecture>

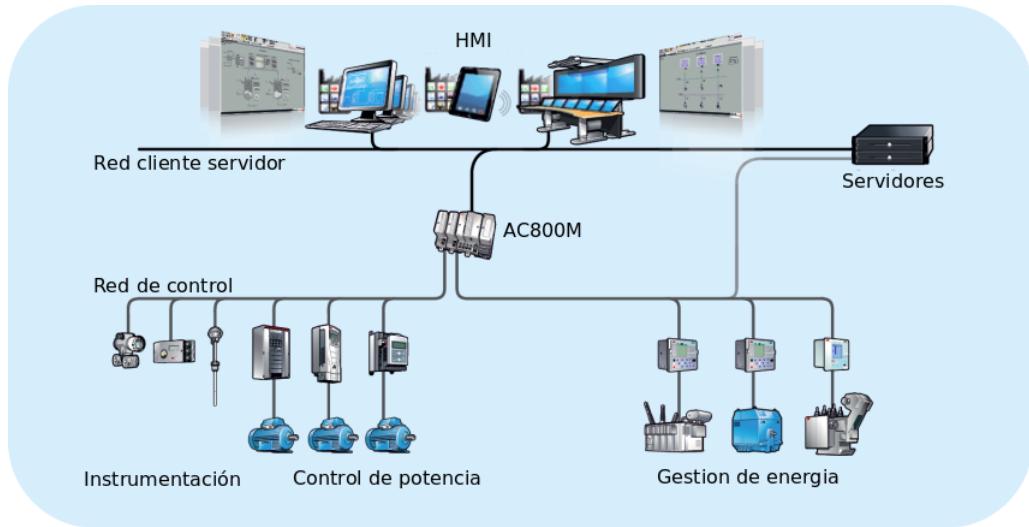


FIGURA 1.3. Sistema de control distribuido 800xA<sup>2</sup>.

El DCS toma decisiones automatizadas basadas en las tendencias de producción de toda la planta. Como ejemplo, el DCS de una planta de energía, podría aumentar automáticamente la capacidad de generación de vapor de múltiples turbinas, para adecuarse a la demanda cambiante de electricidad. Mientras que un PLC puede ajustar la operación de una sola unidad, el DCS puede realizar ajustes en cada una de las unidades de control que interactúan en una planta.

Como sistema de control en este trabajo, se utilizó la herramienta de ABB denominada *SoftController* [7]. Esta nos permite simular la presencia de un controlador físico conectado a la red, con la IP del localhost. El CBM (*Compact Control Builder*) [8], permite crear lógicas del mismo tipo que las usadas en los sistemas DCS 800xA de ABB. De hecho las mismas podrían exportarse al sistema 800xA sin problemas. A continuación se puede ver la figura 1.4 del *SoftController* simulando un controlador.

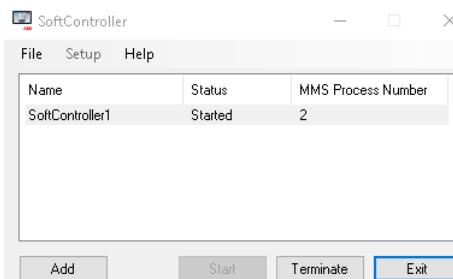


FIGURA 1.4. Controlador simulado<sup>3</sup>.

Por último tenemos el OPC server. Ésta aplicación crea un OPC server para acceder a las variables del sistema de control. Para conectarnos debemos apuntar a la IP del controlador simulado como se muestra en la figura 1.5.

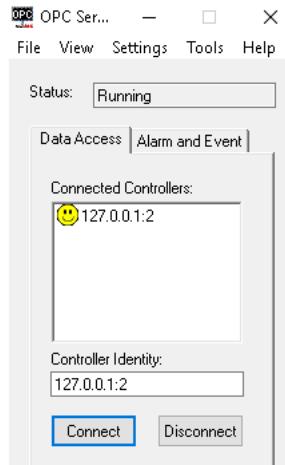


FIGURA 1.5. OPC server<sup>4</sup>.

## 1.5. Motivación

El propósito de este trabajo es innovar en la interacción entre los operadores y los sistemas de control distribuidos, para impulsar nuevas soluciones en el área de la automatización industrial. Se busca explorar las oportunidades que ofrece la realidad aumentada para mejorar y optimizar las tareas de los operadores, además de agilizar el entrenamiento de nuevos operarios y mejorar la seguridad para procedimientos bajo situaciones de emergencia.

## 1.6. Alcance

En la presente memoria, se documenta el trabajo realizado donde se incluyen los siguientes aspectos:

- Desarrollo de la aplicación de realidad aumentada.
- Desarrollo del conector OPC.
- Desarrollo de API REST.
- Comunicación entre el DCS y la interfaz holográfica.
- Implementación de base de datos cloud.



## Capítulo 2

# Introducción específica

En este capítulo se presentan las herramientas usadas para desarrollar el trabajo y un breve compendio de otros trabajos similares en la temática. Hacia el final, se encuentran los requerimientos y la planificación utilizada.

### 2.1. Entorno de desarrollo

#### 2.1.1. Hololens2

El equipo que se utilizó para la interfaz de realidad aumentada es el Microsoft Hololens 2 [9] que se ilustra en la figura 2.1:



FIGURA 2.1. Hololens 2<sup>1</sup>.

Fue lanzado al mercado para uso corporativo y no para el público en general. Microsoft apuesta a que la tecnología sea usada con fines productivos en distintas industrias. Es un equipo de última generación que funciona con un sistema operativo Windows 10 core. La tabla 2.1, muestra sus especificaciones técnicas.

Lo innovador de su diseño es que utiliza microespejos para proyectar tres haces de luz láser de color rojo, verde y azul. Con un microespejo por cada ojo oscilando a 12.000 veces por segundo compone la imagen horizontal. Con un segundo

---

<sup>1</sup>Imagen tomada de <https://img-prod-cms-rt-microsoft-com.akamaized.net/cms/api/am/imageFileData/>

microespejo que oscila 120 veces por segundo forma la componente vertical. Luego el haz de luz es proyectado hacia la retina del usuario, utilizando el visor del Hololens 2 que actúa como una guía de onda.

TABLA 2.1. Especificaciones técnicas Hololens 2

Hardware	Descripción
Procesador	Qualcomm Snapdragon 850
Memoria	4 GB DRAM
Almacenamiento	64 GB
Cámara	1080p
Peso	566 gr
CPU holográfica	Segunda generación
Resolución	2k 3:2
Unidad de medida inercial	Acelerómetro, giroscopio y magnetómetro
Seguimiento de cabeza	4 cámaras de luz visible
Seguimiento de ojos	2 cámaras de infrarrojos

En la figura 2.2 se puede ver una ejemplificación del funcionamiento interno.

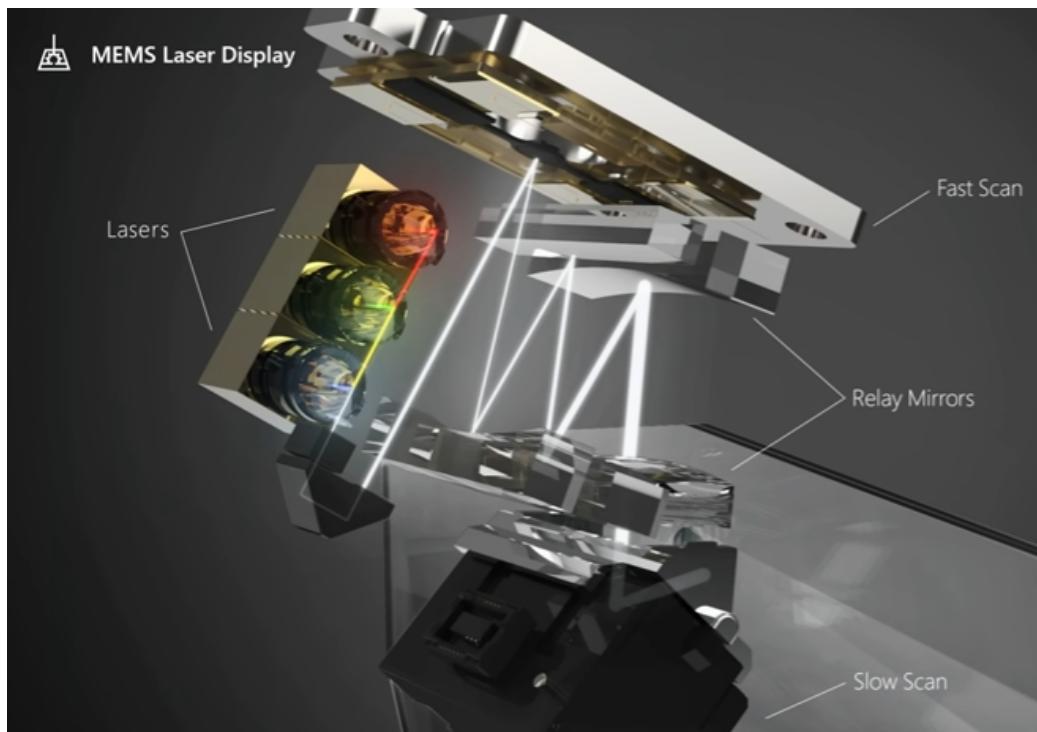


FIGURA 2.2. Microespejos<sup>2</sup>.

### 2.1.2. MRTK

MRTK (*Mixed Reality Toolkit*) para Unity es un kit de desarrollo multiplataforma de código abierto para aplicaciones de realidad mixta [10]. Proporciona un sistema de entrada, componentes fundamentales y bloques de construcción comunes

<sup>2</sup>Imagen tomada de <https://img-prod-cms-rt.microsoft.com.akamaized.net/cms/api/am/imageFileData/>

para interacciones espaciales. Tiene como objetivo acelerar el desarrollo de aplicaciones para Microsoft HoloLens 2, los visores inmersivos de Windows Mixed Reality y la plataforma OpenVR. El proyecto busca reducir las barreras de entrada, crear aplicaciones de realidad mixta y contribuir a la comunidad. El *framework* puede desglosarse conceptualmente como se muestra en la 2.3.

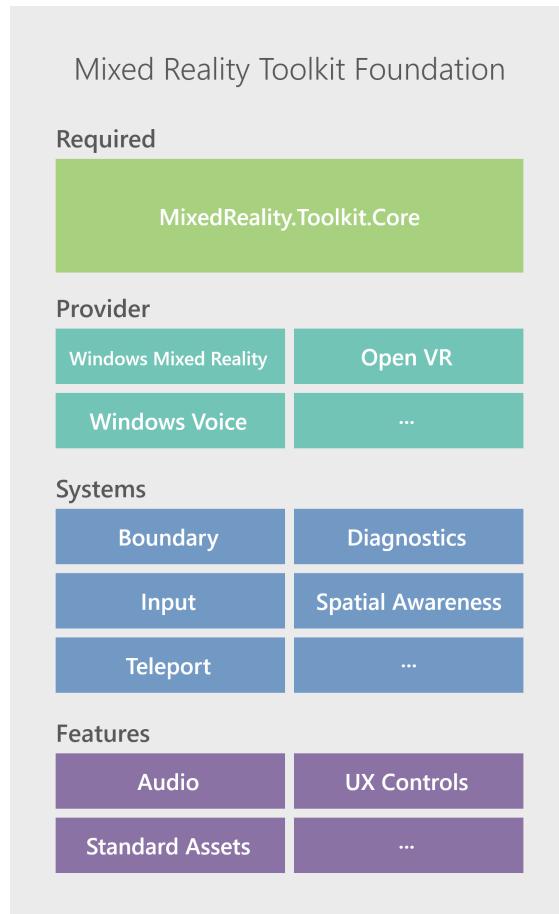


FIGURA 2.3. Componentes del MRTK *framework*<sup>3</sup>.

Los componentes principales del MRTK son:

- *Hands*: clase básica de soporte con servicios para seguimiento de las manos.
- *ObjectMeshObserver*: procesamiento del ambiente usando el modelado 3D.
- *WindowsMixedReality*: compatibilidad con dispositivos *Windows Mixed Reality*, incluidos Microsoft HoloLens y auriculares inmersivos.
- *Profiles*: perfiles predeterminados para los sistemas y servicios de *Microsoft Mixed Reality Toolkit*.
- *SceneSystem*: sistema que proporciona compatibilidad con aplicaciones de múltiples escenas.
- *StandardAssets*: renderizado estándar, materiales básicos y otros activos para experiencias de realidad mixta.

<sup>3</sup>Imagen tomada de <https://docs.microsoft.com/en-us/windows/mixed-reality/>

Además del MRTK se utilizan programas de diseño como el Fusion 360 [11] para crear objetos 3D. Se utiliza el Visual Studio [12] como IDE de desarrollo y como herramienta de compilación. Finalmente se utiliza Unity, como herramienta de diseño visual de la aplicación. En la figura 2.4, se puede ver un diagrama del entorno completo.

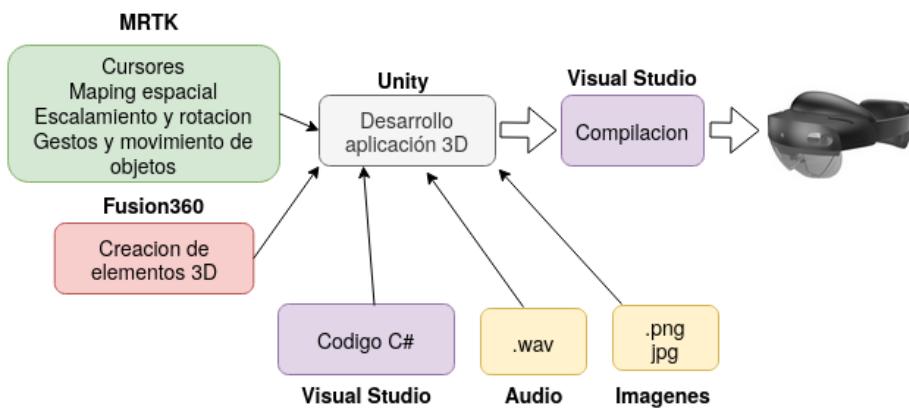


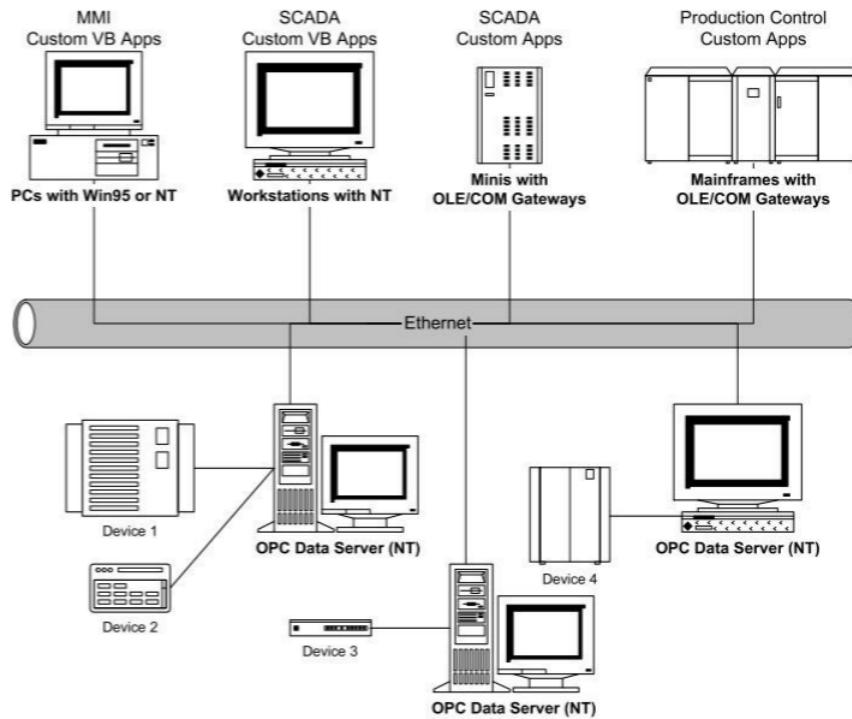
FIGURA 2.4. Entorno de desarrollo<sup>4</sup>.

### 2.1.3. OPC *framework*

OPC (*Open Platform Communications*) es el estándar de interoperabilidad para el intercambio seguro y confiable de datos en la industria [13]. Tiene especial uso en los DCS, dado que son sistemas compuestos por productos de distintos rubros y fabricantes. El protocolo es independiente de la plataforma y garantiza el flujo continuo de información entre dispositivos de múltiples proveedores. La Fundación OPC es responsable del desarrollo y mantenimiento de este estándar. Estas especificaciones definen el acceso a datos en tiempo real, monitoreo de alarmas y eventos, acceso a datos históricos y otras aplicaciones.

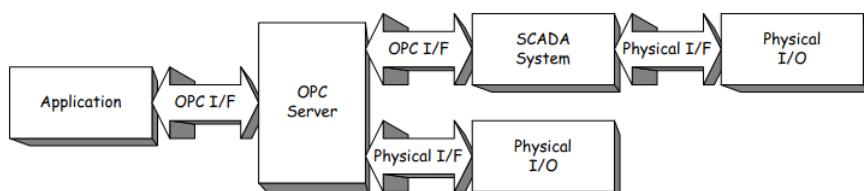
Las especificaciones de OPC *Classic* se basan en la tecnología de Microsoft Windows, utilizando COM / DCOM (*Distributed Component Object Model*) para la comunicación entre componentes de software en una red distribuida cliente-servidor [14]. La especificación original es OPC-DA (*Data Access*), que define una interfaz entre las aplicaciones cliente y servidor para intercambiar datos de proceso y fabricación. Antes de la creación de OPC los productos de distintos proveedores (PLC, HMI) requerían que cualquier dispositivo o aplicación que se conectara a ellos tuviera un “controlador personalizado” que comunicara el dispositivo con equipos de terceros. En la figura 2.5 se muestra una arquitectura típica de control previa a la existencia del standard OPC.

<sup>5</sup>Imagen tomada de <https://technosoftware.com/opc-daaehda-client-net>

FIGURA 2.5. Arquitectura OPC<sup>5</sup>.

Hubo muchos problemas asociados con las comunicaciones basadas en controladores personalizados. La tecnología patentada de alto costo forzaba a los usuarios a mantener un proveedor en particular. Además había dificultades para configurar y mantener actualizados los *drivers* debido al lanzamiento de nuevos dispositivos y aplicaciones. OPC-DA hizo posible la conexión a cualquier fuente de datos en tiempo real sin un *driver* escrito específicamente para el par fuente de datos / receptor de datos. Por lo tanto, las lecturas y escrituras se pueden realizar sin que el receptor de datos tenga que conocer el protocolo nativo de la fuente de datos o la estructura de datos interna.

La especificación OPC describe los objetos y sus interfaces, las cuales son implementadas por los servidores OPC. Aunque OPC está diseñado principalmente para acceder a datos desde un servidor en red, las interfaces se pueden utilizar en muchos lugares dentro de una aplicación. En el nivel más bajo, pueden obtener datos sin procesar de los dispositivos físicos y enviarlos a un SCADA o DCS, o desde el sistema SCADA o DCS a la aplicación. La arquitectura y el diseño permiten que una aplicación cliente acceda a datos de muchos servidores OPC proporcionados por distintos proveedores. En la figura 2.6 se muestra un diagrama en bloques para exemplificar lo explicado.

FIGURA 2.6. Diagrama en bloques<sup>6</sup>.

A alto nivel, un servidor OPC se compone de varios objetos: el servidor, el grupo y el elemento. El objeto del servidor OPC mantiene información sobre el servidor y sirve como contenedor para los objetos del grupo. El objeto del grupo mantiene información sobre sí mismo y proporciona el mecanismo para contener y organizar lógicamente los elementos OPC. Un cliente se comunica con el servidor a través de las interfaces para cada objeto OPC, como se ve en la figura 2.7.

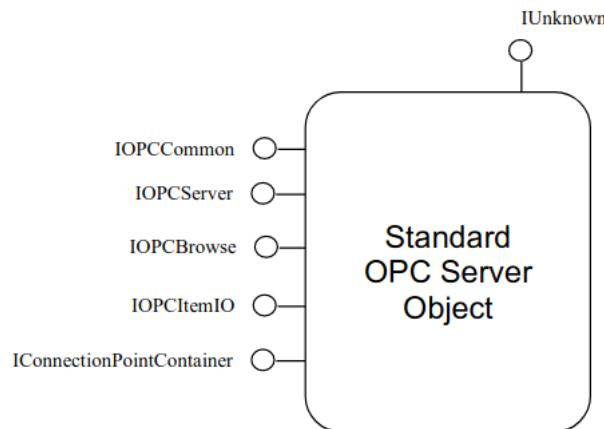


FIGURA 2.7. Objeto OPC<sup>7</sup>.

A continuación se describen brevemente las interfaces del objeto OPC:

- IOPC Server: ésta es la interfaz principal de un servidor OPC, permite realizar las consultas a los grupos de elementos OPC.
- IUnknown: consiste en un puntero a una tabla que permite descubrir dinámicamente si un objeto admite una interfaz en particular.
- IOPC Common:Proporciona la capacidad de configurar y consultar el LocaleID para la sesión cliente/servidor en particular. El LocaleID contempla el formato de números, fechas, moneda, etc. Puede depender de la configuración regional.
- IOPC Browse: proporciona métodos mejorados para navegar por el espacio de direcciones del servidor y para obtener propiedades de los elementos OPC.
- IConnectionPointContainer: permite disparar eventos llamando a un método de la interfaz de un objeto COM del lado del cliente implementado por el mismo cliente.
- IOPC ItemIO: en términos de rendimiento esta interfaz se comportará como si se creara un grupo, se agregarán los elementos OPC, se realizará una sola lectura o escritura y luego se eliminará el grupo. Por lo tanto no es la interfaz recomendada para la mayoría de las aplicaciones.

Los grupos OPC proporcionan una forma para que los clientes organicen los datos. Por ejemplo, el grupo puede representar elementos en una pantalla de operación o variables de un lazo PID. Los datos se pueden leer y escribir. Los elementos

<sup>6</sup>Imagen tomada de <https://opcfoundation.org>

<sup>7</sup>Imagen tomada de <https://opcfoundation.org>

OPC representan conexiones a fuentes de datos dentro del servidor. Un elemento OPC, desde la perspectiva de la interfaz, no es accesible como objeto por un cliente OPC. Por lo tanto, no hay una interfaz externa definida para un elemento OPC.

Todo el acceso a los elementos OPC se realiza a través de un objeto de grupo que contiene los elementos. Un grupo también proporciona una forma para que el cliente se “suscriba” a la lista de elementos para que pueda ser notificado cuando cambien. Este ratio de actualización es un parámetro configurable del servidor.

## 2.2. Requerimientos

A continuación se listan los requerimientos en base a las distintas etapas del trabajo:

1. Requerimientos asociados al desarrollo de la interfaz visual:
  - a) La interfaz debe ser intuitiva y simple.
  - b) El idioma definido es Español.
2. Requerimientos asociados al desarrollo de lógica en .NET:
  - a) La aplicación debe ser fluida y responder sin demoras apreciables por el operador, estableciéndose así el límite máximo de espera en 2 segundos.
  - b) La aplicación debe poder hacer operaciones GET y POST sobre un servidor web, ya sea local o en la nube.
3. Requerimientos asociados a la API rest:
  - a) La API no será de acceso público, sólo podrá ser consultada por las aplicaciones que poseen un *token* de seguridad.
4. Requerimientos asociados a la interfaz de comunicación con el sistema de control:
  - a) La solución debe poder consultar una serie de datos específicos a elección, de los elementos que pertenecen al sistema de control.
  - b) La comunicación debe cumplir con el *standard* OPC.

## 2.3. Planificación

El trabajo fue dividido en tareas y se estimaron los tiempos que debían emplearse para cada una de ellas. A su vez, se analizaron qué tareas debían realizarse primero y cuáles eran sus dependencias. Al momento de organizar las tareas se consideraron 15 horas semanales efectivas de trabajo, obteniendo como resultado la planificación de la figura 2.8:

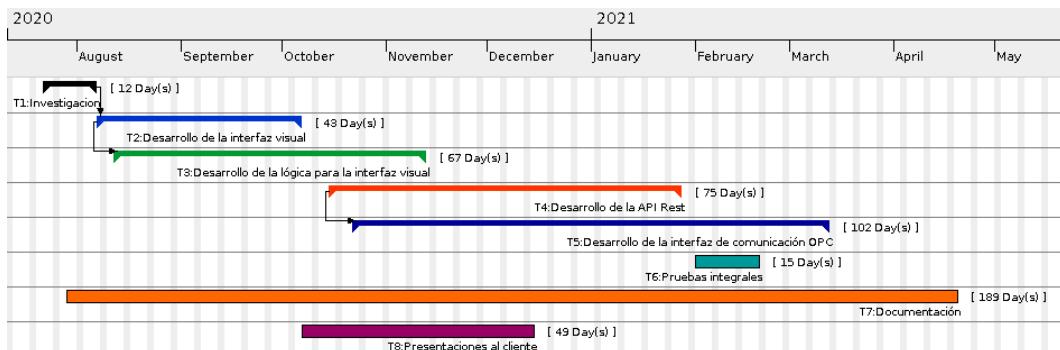


FIGURA 2.8. Diagrama de Gantt<sup>8</sup>.

Como se estimó al inicio del trabajo, la comunicación OPC fue la que más tiempo empleó, incluso se extendió un mes más de lo planeado.

## Capítulo 3

# Diseño e implementación

### 3.1. Análisis del software

#### 3.1.1. Arquitectura del sistema

La solución está integrada por seis componentes principales:

- Interfaz visual: diseñada en Unity3D, le permite al usuario interactuar con la aplicación.
- Lógica .NET: es el backend de la interfaz visual, es la aplicación desarrollada en .NET utilizando CSharp como lenguaje de programación.
- APIrest: a través de ella la aplicación embebida en el Hololens 2 se comunica con el servidor web.
- Base de datos: es actualizada a través de la APIrest y almacena los parámetros del proceso *batch*.
- Comunicación OPC: envía los datos pertinentes al sistema de control a través del *standard* industrial OPC.
- Sistema de control: es representado por un controlador simulado.

En la figura 3.1 se puede observar el diagrama en bloques y la interacción entre los seis componentes principales.

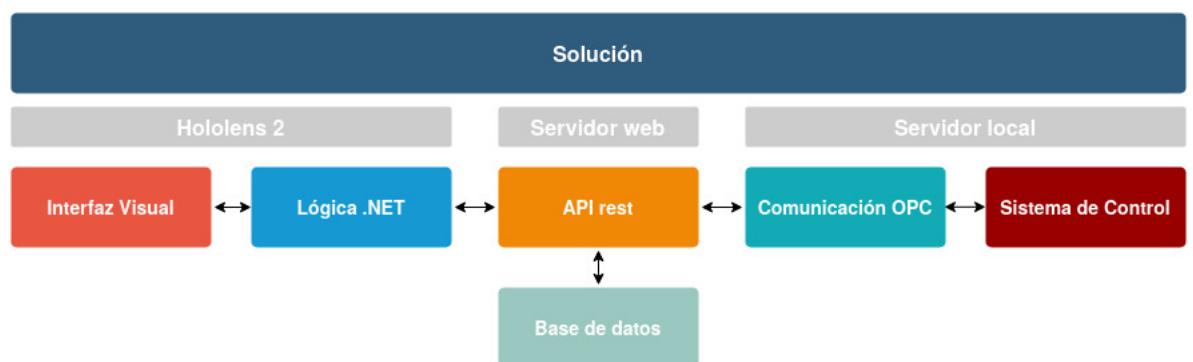


FIGURA 3.1. Diagrama en bloques del sistema<sup>1</sup>.

### 3.1.2. Interfaz visual y lógica .NET

La aplicación embebida en el Hololens 2 se encarga de guiar al operador a través de todo el procedimiento *batch*. En la figura 3.2 se puede ver el diagrama de flujo.

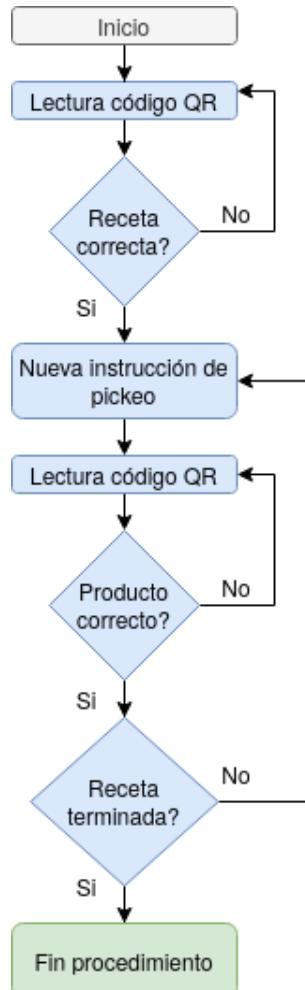


FIGURA 3.2. Diagrama de flujo<sup>2</sup>.

En términos generales la aplicación se basa en la lectura de códigos QR para poder validar la operatoria del usuario. De esta manera, a lo largo del procedimiento se le solicitarán al operador distintas tareas las cuales tendrán asociadas un código QR específico. Por ejemplo, para el agregado de un insumo al tanque de producción, se solicitará la lectura del código de la bolsa del insumo correcto para poder continuar. Mientras que en un proceso de transvase entre tanques, se le solicitará la lectura del código de la válvula que debe actuar para poder continuar. Como se muestra en la figura 3.2, mientras la lectura no sea correcta el operador no podrá avanzar de etapa.

A más bajo nivel, cuando la aplicación inicia se instancia el objeto QRwatcher y el objeto Post. QRwatcher se ejecuta en *background* para detectar la presencia de códigos QR en el ambiente, utilizando la cámara frontal del Hololens 2. Además crea una interfaz de servicio, para que el objeto Post se registre al mismo y reciba un *handler* cuando ocurre una detección. El objeto Post se encarga de realizar las consultas a la API y crea el objeto Navegacion, este actualiza la información mostrada al operador durante todo el procedimiento.

Luego de una detección, el *handler* devuelve el dato codificado de la imagen QR y las coordenadas espaciales del código leído. Utilizando estas coordenadas se instancia un elemento holográfico, para indicar al operador si el código leído es correcto o incorrecto. Para esto se utiliza un tilde verde o una cruz roja sobre el código leído. De acuerdo con el tipo de lectura se dinamizan también botones alrededor del código.

Utilizando los botones holográficos el operador puede cancelar la lectura o aceptarla. Si el código es incorrecto sólo podrá cancelar la lectura. Los botones tienen asociados eventos que disparan *callbacks* al detectarse el pulsado. Si la lectura es correcta, el operador oprimirá aceptar y el *callback* destruirá las instancias holográficas creadas, haciendo desaparecer los botones de la visual del operador. Al mismo tiempo, incrementa la variable de referencia *steps*. Esta variable es pública del objeto Navegacion y se utiliza para determinar en que paso se encuentra la receta. El objeto Navegacion internamente implementa una maquina de estados para indicarle al operador las instrucciones correspondientes en cada etapa.

En la etapa inicial, el objeto Post se encarga de ejecutar una consulta a la API y traer todos los parámetros de la receta leída. Estos parámetros se guardan en un objeto dataReceta el cual permanece inmutable durante toda la aplicación. Con estos datos, el objeto Navegacion es capaz de solicitar las instrucciones correctas en cada etapa e identificar cual sería el código requerido en cada una para poder avanzar. A continuación se muestra la maquina de estados del objeto Navegacion:

```

1  switch (steps)
2  {
3      case 1:
4          textoinstrucion.GetComponent<TMPro.TextMeshProUGUI>().text = POST.
5          dataReceta.msg1;
6          break;
7
8      case 2:
9          _ = POST.PatchAsync("ValidationS1", "1");
10         textoinstrucion.GetComponent<TMPro.TextMeshProUGUI>().text = POST.
11         dataReceta.msg2;
12         break;
13
14     case 3:
15         _ = POST.PatchAsync("ValidationS2", "1");
16         textoinstrucion.GetComponent<TMPro.TextMeshProUGUI>().text = POST.
17         dataReceta.msg3;
18         break;
19
20     case 4:
21         _ = POST.PatchAsync("ValidationS3", "1");
22         textoinstrucion.GetComponent<TMPro.TextMeshProUGUI>().text = POST.
23         dataReceta.msgEnd;
24         ButtonDatos.SetActive(false);
25         canvas_proc.SetActive(false);
26         ButtonMostrar.SetActive(false);
27         MsgFin.SetActive(true);
28         break;
29
30     default:
31         break;
32 }
```

CÓDIGO 3.1. Maquina de estados

La ventaja de esta solución es que el usuario puede modificar los códigos y las instrucciones requeridas sin interferir en la lógica de la aplicación dado que cada etapa se encuentra parametrizada con la información almacenada en la base de datos.

### 3.1.3. API y base de datos

La API (*Application Programming Interface*) fue desarrollada en .NET y se implementaron las siguientes operaciones:

- GET: devuelve los datos de una receta.
- POST: permite crear nuevas recetas.
- PATCH: permite actualizar ciertos campos de la receta sin alterar el resto.
- DELETE: elimina recetas de la colección.

Se creó una clase “Receta” la cual contiene los siguientes parámetros:

- id: número de identificación de la receta.
- Receta: producto a fabricar.
- Descripción: descripción del producto.
- Status: running o stop, indica el estado de la receta.
- Waitinput: sólo si esta en 1 la receta aguarda input del operador.
- ValidationS1: 1 o 0 dependiendo si se cumplió la etapa del procedimiento.
- ValidationS2: 1 o 0 dependiendo si se cumplió la etapa del procedimiento.
- ValidationS3: 1 o 0 dependiendo si se cumplió la etapa del procedimiento.
- ProductoP1: Nombre del producto de la etapa 1.
- ProductoP2: Nombre del producto de la etapa 2.
- ProductoP3: Nombre del producto de la etapa 3.
- CodigoP1: Código del producto de la etapa 1.
- CodigoP2: Código del producto de la etapa 2.
- CodigoP3: Código del producto de la etapa 3.
- CantidadP1: Cantidad del producto de la etapa 1.
- CantidadP2: Cantidad del producto de la etapa 2.
- CantidadP3: Cantidad del producto de la etapa 3.
- Msg1: Mensaje para la etapa 1.
- Msg2: Mensaje para la etapa 2.
- Msg3: Mensaje para la etapa 3.
- MsgEnd: Mensaje de fin de proceso.

Esta clase también se implementó en la lógica del Hololens 2. De esta manera, se utilizan los métodos GET y PATCH durante los pasos de la interfaz para adquirir o enviar datos a la API. Cuando desde la interfaz se lee un código QR internamente se envía una petición GET al servidor web para traer los datos de la receta y almacenarlos en una instancia de la clase Receta. Si la receta es correcta, se utiliza el mismo ID de receta para enviar las actualizaciones de estado cuando el operador avance una etapa. Esta actualización se hace con el método PATCH. Éste se implementó para modificar sólo los parámetros del input de usuario en la receta y no alterar el resto.

La API se hosteo en Azure, la plataforma *cloud* de Microsoft [15]. Se utilizó el servicio Azure Web Apps [16], que es una plataforma que permite publicar aplicaciones web que se ejecutan en múltiples *frameworks* y escritas en diferentes lenguajes de programación. Esto permitió aumentar la velocidad de desarrollo de la aplicación dado que la sincronización y el *deployment* son prácticamente instantáneos. Además, ofrece múltiples posibilidades de integración con otros servicios a futuro.

El motor de la base de datos utilizado es MongoDB [17]. Es una base de datos NoSQL basada en documentos. Se optó por esta tecnología dado que los parámetros de las recetas podían ser encapsulados en un JSON. Se utilizó la plataforma SaaS de MongoDB, Atlas [18], para poder brindar flexibilidad y escalabilidad a la solución con una base de datos cloud. Para acceder a la misma se debe utilizar *usuario* y *password*. La API es la única interfaz que posee esta información, por lo tanto puede realizar las operaciones CRUD [19] sobre la base de datos. A continuación se listan las operaciones:

- *Create*: agrega nuevos documentos a la base de datos.
- *Read*: lee documentos de la base de datos.
- *Update*: actualiza los documentos existentes en la base de datos.
- *Delete*: borra documentos de la base de datos.

En la figura 3.3 se puede ver el método de autenticación elegido en Atlas, para realizar las operaciones CRUD sobre la base de datos.

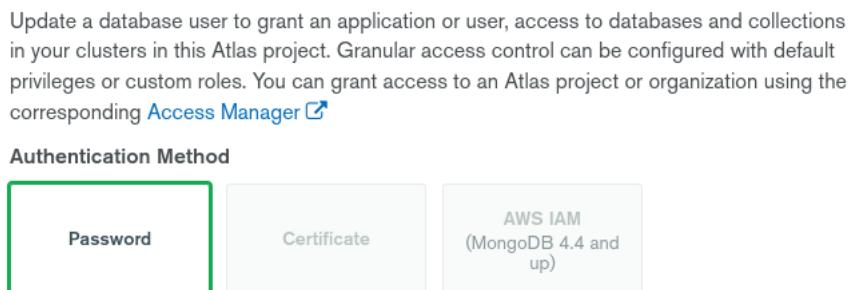


FIGURA 3.3. Método de autenticación elegido<sup>3</sup>.

#### 3.1.4. Comunicación OPC

Para la comunicación OPC se desarrollo un servicio y se instaló en el mismo nodo virtualizado en donde se ejecuta el sistema de control. Se encarga de actualizar

la base de datos según los cambios en el sistema de control. Y viceversa, también es capaz de actualizar las variables de control de acuerdo a la base de datos. Se desarrolló en .NET y se ejecuta como un servicio más de Windows. El servicio incluye el *framework* OPC mencionado en el Capítulo 2, lo que le permite leer y escribir variables OPC. A continuación podemos un extracto de código de la implementación de la interfaz OPC:

```

1 public interface IOpcClient : IDisposable
2 {
3     bool RefreshGruopAfterCreate { get; set; }
4     bool LogSubscriptionsAndWrites { get; set; }
5     bool IsServerRunning { get; }
6     event ServerShutdownHandler OnServerShutdown;
7     void Connect(bool createSubscription);
8     void Connect();
9     void Disconnect();
10    object ReadObject(string idItem);
11    string ReadString(string idItem);
12    ushort ReadWord(string idItem);
13    void WriteString(string idItem, string value);
14    void WriteWord(string idItem, ushort value);
15 }
```

CÓDIGO 3.2. Implementación de la interfaz OPC

Podemos ver las funciones principales para leer y escribir, datos booleanos, strings y enteros del servidor OPC del sistema de control.

Una vez que el servicio se encuentra corriendo se encarga de realizar consultas GET a la APIrest de cada una de las recetas en ejecución. De ésta manera es capaz de detectar que los insumos ya fueron vertidos en el tanque y avanzar así a la siguiente etapa. Además, se encarga de setear la variable “waitinput”, la misma es clave en el funcionamiento de la aplicación dado que indica que la receta se encuentra en ejecución y aguarda órdenes del operador. De ésta manera se valida que el operador sólo pueda iniciar la aplicación con la receta en ejecución por más que intente leer otros códigos de receta inválidos.

### 3.1.5. Sistema de control

Este nodo se implementó en una máquina virtual usando VirtualBox, posee un sistema operativo Windows 10 con 4 gb de memoria RAM y 60 Gb de disco. La ventaja radica en la portabilidad del nodo, dado que podría integrarse a un servidor ESXI [20] en cualquier entorno industrial.

En el CBM podemos ver los bloques de las distintas recetas que se crearon en la aplicación de control, como se muestra en la figura 3.4.

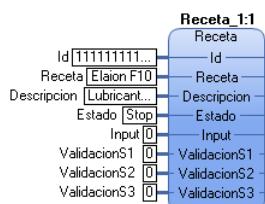


FIGURA 3.4. Bloque receta<sup>4</sup>.

A medida que nuevas recetas sean necesarias, estos bloques funcionan como objetos de una clase que pueden instanciarse para cada tipo de receta. Simultáneamente estas recetas se cargan en la base de datos para establecer la comunicación entre el sistema de control y la interfaz del Hololens 2. Dentro de la receta se puede ver una lógica secuencial del procedimiento, que irá avanzando a medida que el operador complete los pasos en la interfaz holográfica. Podemos ver en la figura 3.5 el avance de la secuencia en el tiempo.

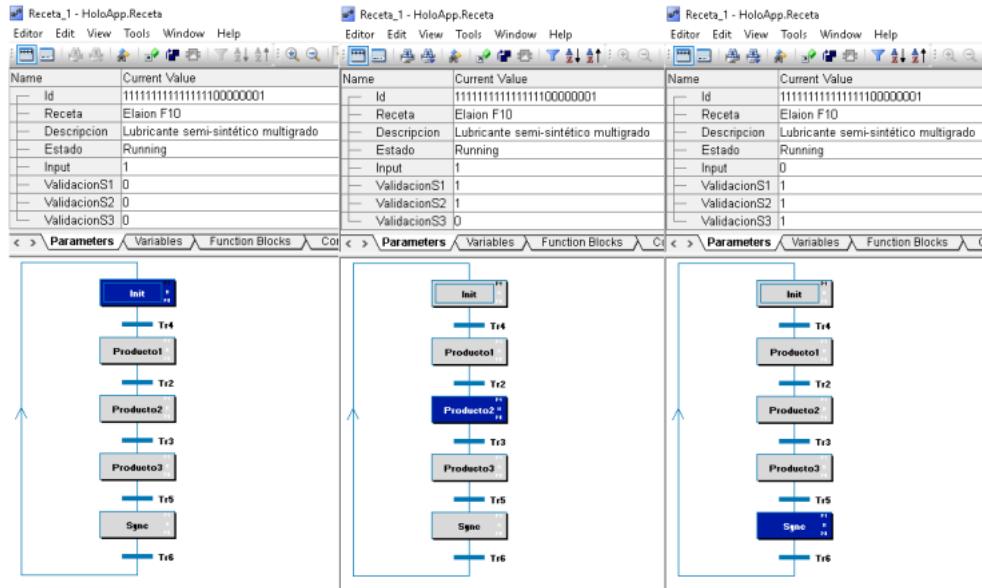


FIGURA 3.5. Secuencia<sup>5</sup>.

La solución propuesta es fácilmente escalable desde la lógica de control y no esta atada a ningún proveedor en particular. Esto se debe a que en cualquier sistema de control somos capaces de crear este clase de objetos y comunicarlos vía el *standard OPC*. Es por eso que el servicio OPC desarrollado, es capaz de operar con distintos sistemas de control sin alterar el funcionamiento de la aplicación desarrollada para el Hololens 2.



## Capítulo 4

# Ensayos y resultados

### 4.1. Aplicación

La aplicación se encarga de ayudar a los operadores durante la fabricación por lotes de un producto determinado. Durante los procedimientos *batch* se utilizan el mismo conjunto de máquinas para la fabricación de distintos productos. Por lo tanto, las tareas de los operadores varían en las distintas partes del proceso dependiendo del producto que se esté fabricando.

Una etapa clave en la fabricación del producto, es el agregado de aditivos en la mezcla. La aplicación desarrollada se encarga de evitar que el operador falle a la hora de seleccionar y dosificar el aditivo requerido. La aplicación comienza con la imagen que se muestra en la figura 4.1.



FIGURA 4.1. Pantalla inicial<sup>1</sup>.

En este punto se aguarda la lectura del código QR correspondiente a la receta a producir. Las recetas son el conjunto de parámetros que involucran un lote de producción. En ellas se puede encontrar el listado de aditivos, las cantidades requeridas, los códigos de los distintos productos, tiempos de mezclado y demás parámetros necesarios para la producción. Si el operador lee un código de receta para el cual el sistema de control no fue configurado a producir en ese momento, se indicará un error como muestra la figura 4.2.

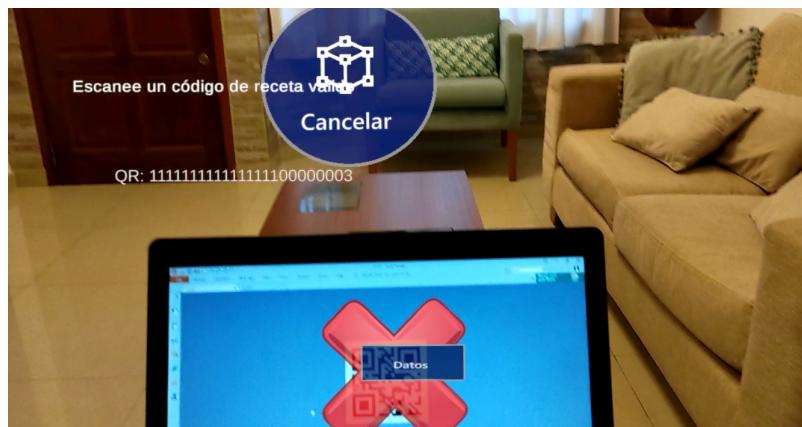


FIGURA 4.2. Código erróneo<sup>2</sup>.

Para poder avanzar, es necesario que se lea el código de la receta correcta, como se muestra en la figura 4.3.

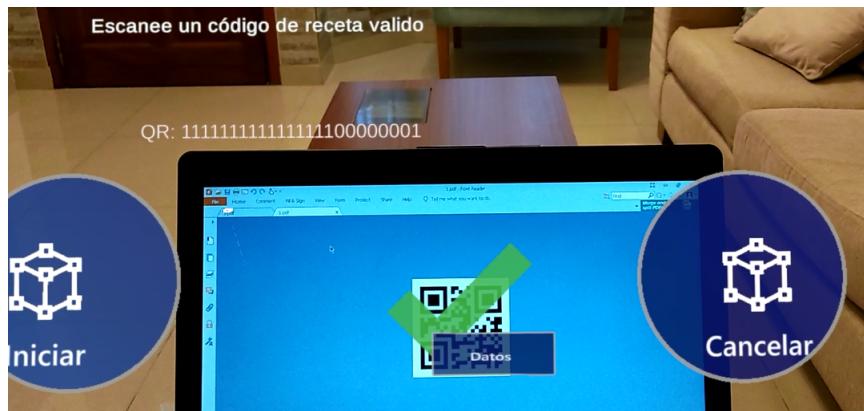


FIGURA 4.3. Código correcto<sup>3</sup>.

En éste punto el operador acepta la receta y la interfaz mostrará la imagen que se presenta en la figura 4.4.



FIGURA 4.4. Inicio de receta<sup>4</sup>.

En el margen derecho se pueden ver tres círculos azules, los cuales representan tres tareas del operador a lo largo de la receta de producción. En la parte superior del listado de tareas se puede leer el producto que se fabricará en esta receta.

En el lado izquierdo superior se puede ver una leyenda que irá actualizándose para guiar al operador. En la parte inferior se encuentran cuatro botones con las siguientes funcionalidades:

- Ocultar: permite simplificar la vista holográfica, reduciendo la información proyectada en el Hololens 2.
- Mostrar: restablece la información holográfica oculta.
- Datos: muestra los datos del código QR leído.
- Siguiente: se utiliza para avanzar a la siguiente etapa.

Según lo indicado en pantalla el operador deberá pulsar “Siguiente” para iniciar el procedimiento como se muestra en la figura 4.4. En ese momento la instrucción se modificará, solicitándole que vierta en el tanque de producción el producto A. Esto se ilustra en la imagen de la figura 4.5.



FIGURA 4.5. Selección insumo<sup>5</sup>.

El operador deberá localizar las bolsas de éste producto y leerá con el Hololens 2 el código QR de la misma. En ese momento el conjunto de datos se actualizará como se ilustra en la imagen de la figura 4.6.

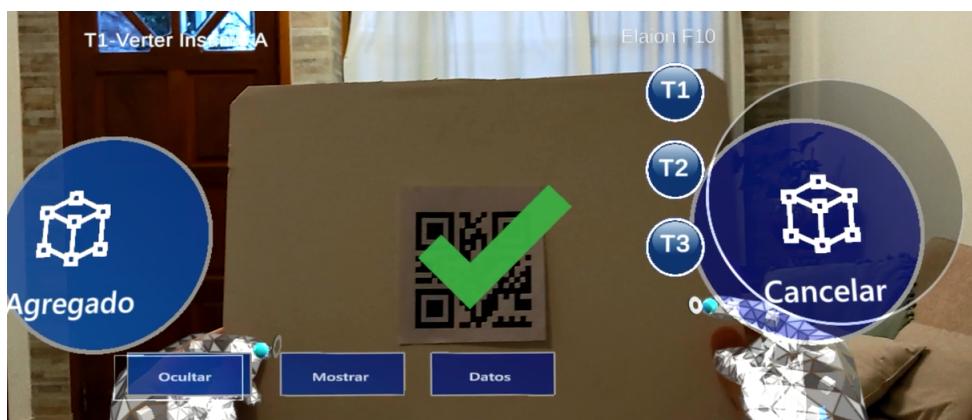


FIGURA 4.6. Insumo correcto<sup>6</sup>.

Si el producto es correcto se mostrará un tilde verde, de lo contrario se mostrará una cruz roja como en la figura 4.2. El operador pulsara “Agregado” y luego deberá verter las cantidades indicadas del producto en el tanque de producción.

En el centro izquierdo de la imagen, se pueden ver los datos que hacen referencia al aditivo que se usará en la receta. Como se ilustra en la imagen de la figura 4.7.



FIGURA 4.7. Producto elegido<sup>7</sup>.

Se muestran los siguientes datos:

- QR: código QR leído por el Hololens 2.
- Producto: nombre del producto leído.
- Código: código del producto leído.
- Cantidad: cantidad necesaria del producto en la receta.

Luego el operador pulsara “Siguiente” y se actualizará la información con una nueva instrucción como se ilustra en la figura 4.8.



FIGURA 4.8. Selección de producto<sup>8</sup>.

El procedimiento se repetirá para el total de las tareas de la receta y a medida que las tareas se completen se pintaran de color verde. Luego de la última operación, se mostrará la leyenda “Procedimiento terminado 3/3”, como se ilustra en la imagen de la figura 4.9 indicándole al operador que puede cerrar la aplicación.



FIGURA 4.9. Fin de la aplicación<sup>9</sup>.

Los comandos al sistema de control son enviados luego de que el producto correcto fue leído y el operador avanza al siguiente paso de la receta, garantizando que se cumplió la tarea asignada. Si el producto erróneo es leído, el operador no tiene habilitado el paso siguiente, por lo tanto queda retenido en esa etapa. Además, las recetas válidas son sólo aquellas en las que el sistema de control aguarda operaciones manuales, de esta manera se asegura que no puedan mezclarse otras recetas con la receta activa.

## 4.2. Evaluación de las interfaces

A continuación se puede ver algunas mediciones de tiempos de respuesta de las interfaces principales. Para la medición Hololens 2 - API, se utilizó la opción *debug* del Hololens 2 para identificar el *timestamp* del comando emitido al clickear “Siguiente” en la interfaz. Además se utilizó la API para loguear el *timestamp* en el que se recibe la consulta web.

Para la medición API - server OPC, también se utilizó la API para loguear el *timestamp* en el que se recibe la consulta web. Y se utilizó el servicio OPC para loguear el momento en el que la variable OPC es actualizada. Para un promedio de 10 mediciones consecutivas, los resultados pueden verse en la tabla 4.1.

TABLA 4.1. Tiempos de respuesta promedio

Interfaz	Tiempo
Hololens 2 - API	>340ms
API - server OPC	>320ms

Dado que la aplicación se comunica a través de mensajes del tipo JSON y la cantidad de información enviada es reducida, los tiempos de respuesta son realmente bajos y no representan un problema para la implementación. Se puede ver que el tiempo de respuesta de toda la cadena de comunicación es inferior al segundo. Este trabajo se realizó utilizando servidores *cloud* pero bien podrían ser locales.

Llegado el caso, si los tiempos representan una pérdida de fluidez en la aplicación, la migración no requeriría mayores esfuerzos.

### 4.3. Devoluciones del cliente

Las presentaciones realizadas a usuarios industriales resaltaron las fortalezas de la tecnología. Se presentó interés en desarrollar aplicaciones orientadas al mantenimiento de máquinas. Estableciendo una serie de pasos que el operador debe realizar periódicamente para prolongar la vida útil de la máquina. También se mencionó su aplicación en el entrenamiento para el uso de las máquinas. Donde el operador pueda ensayar en primer instancia, sobre una réplica holográfica de la máquina las distintas configuraciones de operación. Y en segunda instancia, asistido por instrucciones holográficas sobre la máquina real. Por último, se identificaron operaciones en planta donde el Hololens 2 podría reducir los tiempos de operación. Ahorrándole al operador la necesidad de consultar los sistemas de producción en busca de órdenes de trabajo, identificaciones de productos o estado de distintos equipos en el sistema de control.

## Capítulo 5

# Conclusiones

### 5.1. Resultados obtenidos

Fue muy importante la planificación inicial para enmarcar y organizar el trabajo. Las estimaciones fueron correctas, aunque se subestimó la etapa de integración de todos los componentes. El trabajo logró integrar realidad aumentada, plataformas *cloud* y sistemas de control tradicionales con éxito. La aplicación resultó útil en las pruebas realizadas, y durante las presentaciones con clientes industriales el *feedback* fue positivo. El trabajo demostró que las integraciones de sistemas de control con las tecnologías 4.0 son posibles, pero es necesario encontrar casos de uso prácticos que justifiquen el esfuerzo del desarrollo. La mano de obra no sólo puede automatizarse, sino que también puede mejorarse con ayuda de las nuevas tecnologías. La realidad aumentada busca mejorar el ritmo de trabajo del operador y al mismo tiempo disminuir la tasa de falla guiando la operación. Ambos son factores claves para optimizar el rendimiento operativo.

### 5.2. Próximos pasos

El siguiente paso es utilizar un *token* para autenticar la comunicación y certificados TLS para encriptar el canal de comunicación entre el Hololens 2 y el servidor web. De esta manera se mejoraría la seguridad considerablemente. Utilizando además lo desarrollado, se tiene el punto de partida para realizar una segunda aplicación orientada al mantenimiento de activos en la planta.



# Bibliografía

- [1] Sara Salman Hassan. *Smart retrofitting of machine tools in the context of industry 4.0*. Disponible: 2021-06-26. 2021. URL: <https://www.sciencedirect.com/science/article/pii/S2212827120303838>.
- [2] Matthias Hebenstreit and Michael Spitzer and Matthias Eder. *An industry 4.0 production workplace enhanced by using mixed reality assembly instructions with microsoft hololens*. Disponible: 2021-06-26. 2021. URL: [https://www.researchgate.net/publication/344375201\\_An\\_Industry\\_40\\_Production\\_Workplace\\_Enhanced\\_by\\_Using\\_Mixed\\_Reality\\_Assembly\\_Instructions\\_with\\_Microsoft\\_HoloLens](https://www.researchgate.net/publication/344375201_An_Industry_40_Production_Workplace_Enhanced_by_Using_Mixed_Reality_Assembly_Instructions_with_Microsoft_HoloLens).
- [3] Matthias Hebenstreit and Michael Spitzer and Matthias Eder. *An investigation of mixed reality technology for onsite construction assembly*. Disponible: 2021-06-26. 2021. URL: [https://www.matec-conferences.org/articles/matecconf/abs/2020/08/matecconf\\_eppm2018\\_06001/matecconf\\_eppm2018\\_06001.html](https://www.matec-conferences.org/articles/matecconf/abs/2020/08/matecconf_eppm2018_06001/matecconf_eppm2018_06001.html).
- [4] Ian Sutton. *Process Risk and Reliability Management*. Gulf Professional Publishing, 2015.
- [5] ABB. *Distributed control systems*. Visitado el 2021-06-26. 2021. URL: <https://new.abb.com/control-systems>.
- [6] ABB. *ABB Ability™ system 800xA architecture*. Disponible: 2021-06-26. 2021. URL: <https://new.abb.com/control-systems/system-800xa/800xa-dcs/system/architecture>.
- [7] ABB. *SoftController*. Disponible: 2021-06-26. 2021. URL: <https://new.abb.com/control-systems/system-800xa/800xa-dcs/hardware-controllers-io/control-builder-engineering-software>.
- [8] ABB. *Compact Control Builder for AC 800M Controller*. Disponible: 2021-06-26. 2021. URL: <https://new.abb.com/control-systems/essential-automation/compact-product-suite/essential-controller-suite/compact-control-builder>.
- [9] Microsoft. *Hololens 2*. Disponible: 2021-06-26. 2021. URL: <https://www.microsoft.com/es-es/hololens>.
- [10] Microsoft. *What is the Mixed Reality Toolkit*. Disponible: 2021-06-26. 2021. URL: <https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05>.
- [11] Autodesk. *Fusion 360*. Disponible: 2021-06-26. 2021. URL: <https://www.autodesk.com/products/fusion-360/overview?term=1-YEAR>.
- [12] Microsoft. *Visual Studio*. Disponible: 2021-06-26. 2021. URL: <https://visualstudio.microsoft.com/es/>.
- [13] OPC foundation. *What is OPC?* Disponible: 2021-06-26. 2021. URL: <https://opcfoundation.org/about/what-is-opc/>.
- [14] OPC foundation. *What is OPC Classic?* Disponible: 2021-06-26. 2021. URL: <https://opcfoundation.org/faq/what-is-opc-classic/>.
- [15] Microsoft. *Azure*. Disponible: 2021-06-26. 2021. URL: <https://azure.microsoft.com/en-us/overview/what-is-azure/>.

- [16] Microsoft. *Azure Web Apps*. Disponible: 2021-06-26. 2021. URL: <https://azure.microsoft.com/en-us/services/app-service/web/>.
- [17] MongoDB. *What is MongoDB?* Disponible: 2021-06-26. 2021. URL: <https://www.mongodb.com/what-is-mongodb>.
- [18] MongoDB. *MongoDB Atlas*. Disponible: 2021-06-26. 2021. URL: <https://www.mongodb.com/cloud/atlas>.
- [19] MongoDB. *MongoDB*. Disponible: 2021-06-26. 2021. URL: <https://docs.mongodb.com/manual/crud/>.
- [20] VMware. *VMware ESXi: The Purpose-Built Bare Metal Hypervisor*. Disponible: 2021-06-26. 2021. URL: <https://www.vmware.com/products/esxi-and-esx.html>.